

## Text to vector

BOW :-

Bag of Words :-

It is a technique to convert a text to vector.

R<sub>1</sub>: This pasta is tasty and delicious

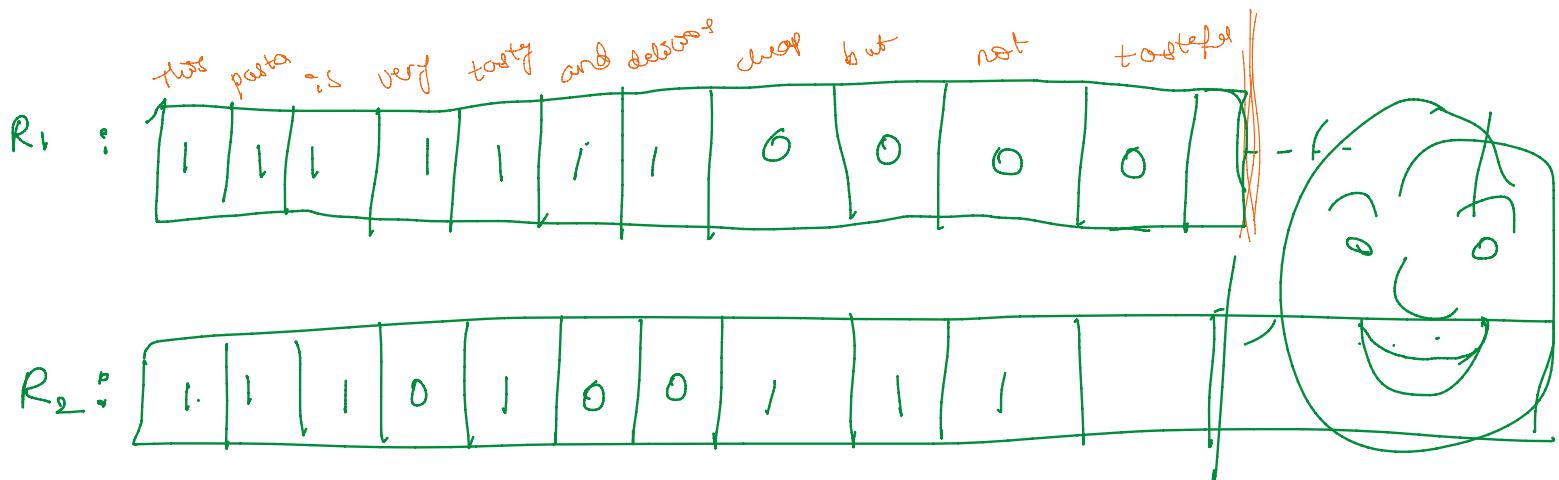
R<sub>2</sub>: This pasta is not tasty but affordable

R<sub>3</sub>: This pasta is delicious and cheap

R<sub>4</sub>: Pasta is tasteful.

Now first we need to form vocabulary of all unique words.

{This, pasta, is, tasty, and, delicious, affordable, but, not,  
cheap, tasteful.}



So we fill the counts of the words in our text to form vector.

Here we consider #counts.

\* For n-gram

here we consider #count.

For binary BOW,

$$\begin{cases} 1 & \text{if } w_i \text{ present} \\ 0 & \text{if } w_i \text{ not present.} \end{cases}$$

\* Here we can see that  $R_1$  &  $R_2$  are opposite, but distance between them is low.

\* Also BOW doesn't take consideration of semantic meaning

Unigram - Bigram - trigram —— n-gram

Previous example is unigram.

If we consider pair of words to construct vector, that is called bigram.

If we consider n-words at a time to construct vector  
 $\longrightarrow$  n-gram

Text Preprocessing steps :-

## D. Stop word Removal

"Not always a better choice"

- ② Convert to lower case
  - ③ Remove special characters
  - ④ Short form → full form
  - ⑤ Stemming
    - \* Snowball stemmer → New.
    - \* Porter stemmer → Old

Converts ,  
Study , studying, studies  
{ studi

No grammar rule

## 6) Lemmatization

## Consideration of Parts of Speech:

go went go <sup>ed</sup>

Lemmatization usually refers to **doing** things properly with the use of a vocabulary and morphological analysis of **words**, normally aiming to remove inflectional endings only and to return the base or dictionary form of a **word**, which is known as the **lemma**.

It means understanding the structure of text, **Morphological analysis** is the process of providing grammatical information about the word on the basis of properties of the morpheme(basic unit that can not be further divided) it contains.

In order to create the BoW firstly, we are required to remove the stop words. Then we process our Stemming process. Stemming is the algorithmic process where each words having the same sense of understanding are converted to its stem or root. Here root can be a word in itself or not. Example:  
study , studying , studies all stemmed to studi (not study). Each word can be formed by adding the suffixes to the root.  
We also perform Lemmatisation which is the algorithmic process of combining the inflected words into a common word called lemma. The Algorithm make use of the

dictionary to guess down the lemma. Suppose we have words = {go , going , went} we get lemma as "go". If you want to know how nltk does it let me give u a rough idea of it.

NLTK makes use of Parts of speech - Tagging , where parts of speech tagger take group of words as an input and it returns list of tuples. Tuple contains a word and corresponding parts of speech.This parts of speech tagging is done to make sense of context in the sentence and helps lemmatizer to choose the appropriate lemma.

## Tokenization

**Tokenization** is basically breaking our paragraphs into list of sentences and sentences into list of words

New York → New\_York

New Delhi → New\_Delhi

After these preprocessing, we apply BOW / any vectorization technique -

though we can't preserve semantic meaning

## TF-IDF :

Term frequency      Inverse - Document frequency :-

Term frequency,

$$TF(w_i^e, r_j^e) = \frac{\# \text{ times } w_i \text{ occurs in } r_j^e}{\text{total } \# \text{ words in } r_j^e}$$

$$0 \leq TF(w_i^e, r_j^e) \leq 1$$

$w_i^e \rightarrow$   $i^{th}$  word

$r_j^e \rightarrow$   $j^{th}$  document

i.e.  $w_i$  is the  $i^{th}$  word in  $j^{th}$  document.

i.e. TF is frequency of word in particular document.

IDF :-

$$IDF(w_i, D_c) = \log\left(\frac{N}{n_i}\right)$$

$N \rightarrow$  total # documents.

$D_c \rightarrow$  text corpus.

$n_i \rightarrow$  # documents in which  $w_i$  occurs.

So IDF is calculated for word  $w_i$  in consideration with total text corpus  $D_c$ .

always,  $\underline{\underline{N \geq n_i}}$

$$\Rightarrow \frac{N}{n_i} \geq 1 \Rightarrow \log\left(\frac{N}{n_i}\right) \geq 0$$
$$\Rightarrow \underline{\underline{IDF(w_i, D_c) \geq 0}}$$

As  $n_i \uparrow_{se} \rightarrow \frac{N}{n_i} \downarrow_{se} \rightarrow \log\left(\frac{N}{n_i}\right) \downarrow_{se}$   
 $\Rightarrow \underline{\underline{IDF(w_i, D_c) \downarrow_{se}}}$

i.e.

IDF value will be low for most repeating words in  $D_c$

As  $w_i$  repeats more in  $D_c$ ,  $\underline{\underline{IDF(w_i, D_c) \downarrow_{se}}}$

$$\text{Wt. of } n_i \text{ doc} \rightarrow \left(\frac{N}{n_i}\right)^{\text{rec}} \rightarrow \log\left(\frac{N}{n_i}\right)^{\text{rec}}$$

$$\text{IDF}(w_i, D) \text{ rec.}$$

i.e. IDF value will be high for rare word in corpus.

$$\text{tfidf}(w_i, r_j) \rightarrow \text{TF}(w_i, r_j) * \text{IDF}(w_i, D_c)$$

$\text{TF} \uparrow \rightarrow$  most frequent word in particular document.

$\text{IDF} \uparrow \rightarrow$  Rare word in total text corpus.

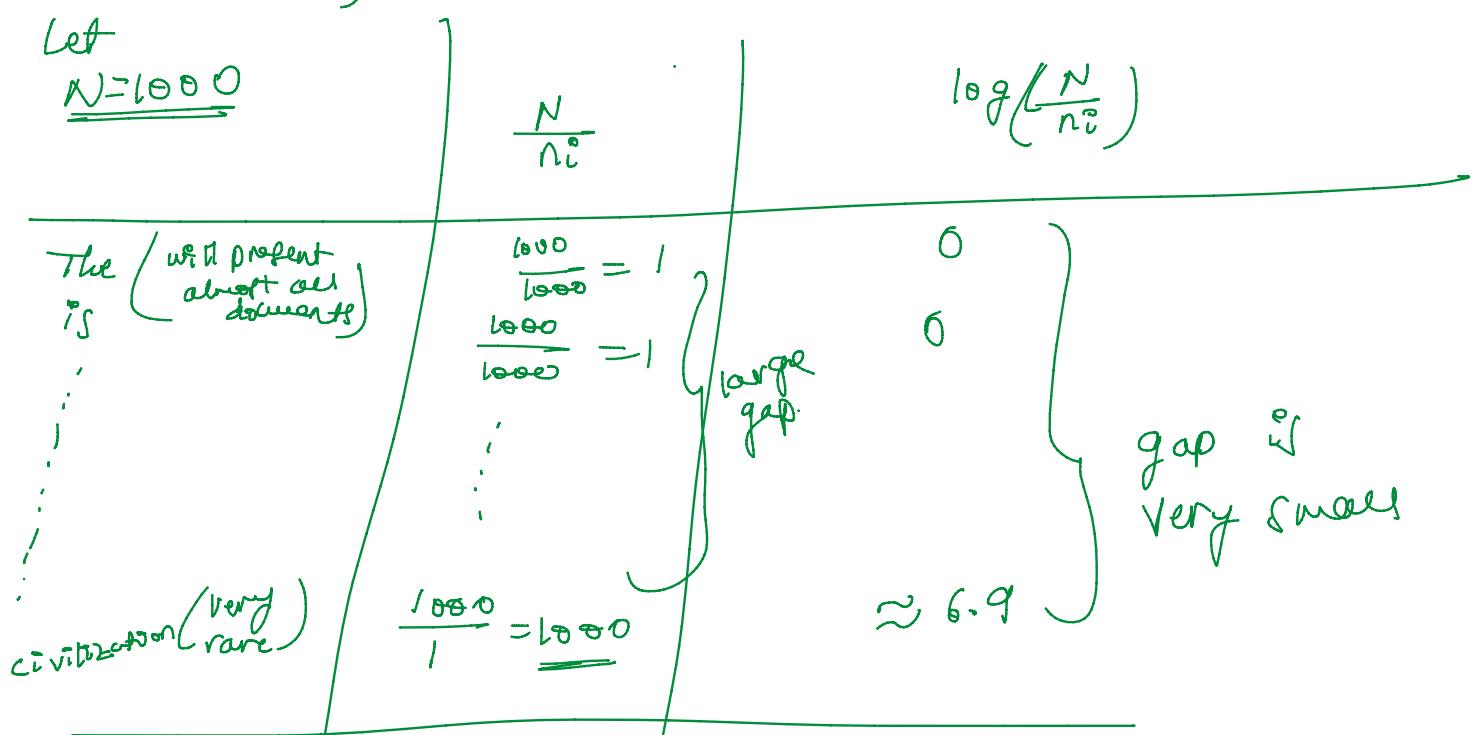
why log in idf ?

there is no strong theory behind this.

Some Intuition,

Let

$$N = 1000$$



$$\text{civ} \cdot \text{w} \cdot \text{rare} = 1 = \frac{N}{n_i}$$

Hence If we don't consider log,

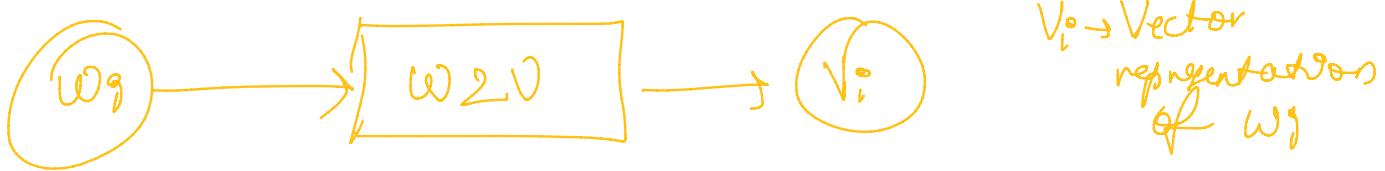
$\frac{N}{n_i}$  value will dominate in tf \* idf  
 Log will scale the values to smaller values

W2V :-

W2V is state of the art technique.

It considers semantic meaning.

Its maths is very rigorous and discussed in Deep Learning



$v_i \rightarrow$  Vector representation of  $w_i$

Google has trained using google news data  
 Its 300-d vectors for each word.  
 (Glove model)

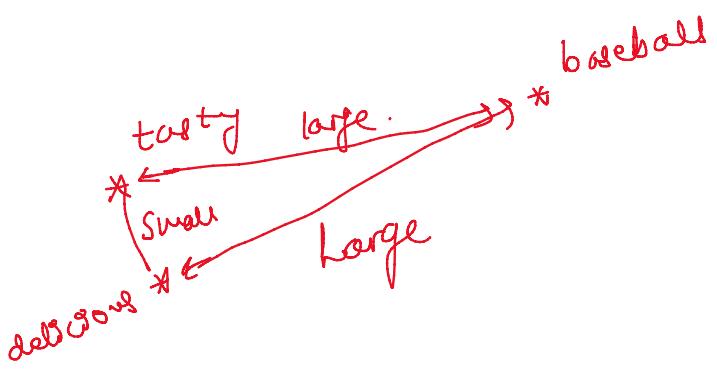
## Glove model

What  $w2v$  does is,

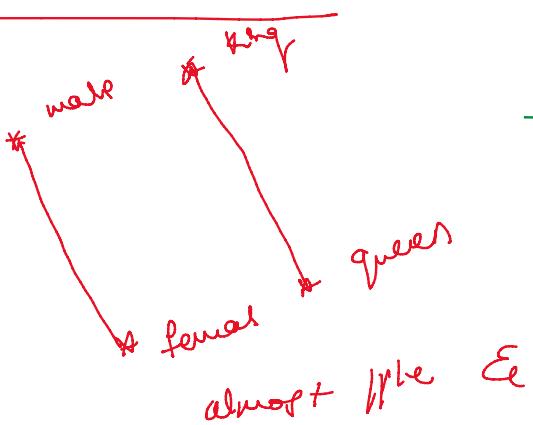
- It preserves semantic meaning.

If words are semantically similar }  
their vectors are closer }

---



- Identifies Relationships.



$$(v_m - v_f) \parallel (v_k - v_q)$$

Why for country - capitals also.

---

Larger the dimension of vector

↳ more information.

## Avg w2v

$R_1 \rightarrow w_1, w_2, w_3, \dots, w_n$

$w_1 \rightarrow [1 | 1 | 1 | \dots] v_1$

$w_2 \rightarrow [1 | 1 | 1 | \dots] v_2$

all  $w_i \in w_1$ .

$$\boxed{\text{Avg } w_2v \rightarrow \frac{\sum_{i=1}^n v_i}{N}}$$

## tf-idf w2v

$$= \frac{t_1 * w_2v(w_1) + t_2 * w_2v(w_2) + t_3 * w_2v(w_3) + \dots + t_n * w_2v(w_n)}{\sum_{i=1}^n t_i}$$

$t_i \rightarrow$  tf-idf value of  $i^{th}$  word.

Avg w2v & tf-idf w2v are weighting schemes.